Paper: GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. [5]

Jiaxin Liu

Group Reading

October 7, 2021
Overview

1. Introduction
   - Over-sampling and under-sampling
   - SMOTE

2. GraphSMOTE
   - Feature extractor and synthetic node generation
   - Edge generator
   - GNN classifier
   - Optimization Objective

3. Experiments
Introduction

- Class imbalance problem
  - Algorithm-level: cost sensitive learning.
  - Data-level: re-sample the original dataset such as SMOTE[1].
  - Hybrid approaches.
Introduction

- Class imbalance problem
  - Algorithm-level: cost sensitive learning.
  - Data-level: re-sample the original dataset such as SMOTE[1].
  - Hybrid approaches.
- Re-sample [3]:
  - Over-sampling: random and focused over-sampling for minority class.
  - Under-sampling: random and focused under-sampling for majority class.

![Figure: An example for re-sampling.](image-url)
Synthetic Minority Over-sampling Technique (SMOTE)

- **Over-sampling:**
  - Replicate the original data.
  - Generate new synthetic data.

- **SMOTE:**
  - Over-sample the minority class.
  - Synthetic examples are introduced along the line segments joining any/all of the k minority class nearest neighbors.

*Figure:* An example for SMOTE.
SMOTE

- Decision region for over-sampling the minority class with replication (left) and synthetic generation (right).

Figure: Decision region (solid line) as a result of replicating minority directly.

Figure: Decision region (dashed line) as a result of using synthetic data.
Table: Comparison between SMOTE[1] and Mixup [4].

<table>
<thead>
<tr>
<th>Source for the generation</th>
<th>SMOTE</th>
<th>Mixup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class for the synthetic data</td>
<td>Minority class</td>
<td>$\lambda y_1 + (1 - \lambda)y_2$</td>
</tr>
<tr>
<td>Model training</td>
<td>Original and synthetic data</td>
<td>Synthetic data only</td>
</tr>
<tr>
<td>Weakness</td>
<td>Know the neighbors’ info</td>
<td>Inaccurate synthetic label</td>
</tr>
</tbody>
</table>
GraphSMOTE

Task: node classification task on graph $G = \{V, A, F\}$ in the transductive setting.

- $V = \{v_1, \cdots, v_n\}$ is a set of $n$ nodes.
- $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix.
- $F \in \mathbb{R}^{n \times d}$ denotes the node attribute matrix.
- $Y \in \mathbb{R}^n$ is the class information for node in $G$.
- $V_L, Y_L$ denotes the nodes in the training set and their labels.
- $m$ classes: $\{C_1, \cdots C_m\}$
- Imbalanced ratio: $\frac{\min_i |C_i|}{\max_i |C_i|}$ statisticized from $V_L$.

Goal: given the imbalanced node class set and a labeled training set $V_L$, find a node classifier $f(V, A, F) \rightarrow Y$ that works well for both majority and minority classes.
GraphSMOTE

Idea:
1. generate synthetic minority nodes → feature encoder and node generator;
2. assign links for these synthetic nodes → edge generator;
3. train the GNN on this augmented balanced graph → GNN classifier.

Figure: An example of bot detection on a social network and the idea of over-sampling.
Feature extractor and synthetic node generation

- Feature extractor?
  - Raw node feature space is sparse and high-dimensional → hard to get similar nodes from the same class
  - Raw features don’t consider the graph structure.
- Use one block of GraphSAGE [2] as the feature extractor.

$$h_v^1 = \sigma(W_1 \cdot \text{CONCAT}(F[v,:], F \cdot A[ :, v]))$$

**Figure:** An example for GraphSAGE.
Node generation

Adopt SMOTE algorithm to generate synthetic node using the embedding features.

For a labeled minority node $h_v^1$ and its label $Y_v$

1. Find closest labeled node to node $h_v^1$ in class $Y_v$.

$$nn(v) = \arg \min_u \| h_u^1 - h_v^1 \|, \quad Y_u = Y_v$$

2. Generate the synthetic node.

$$h_v^{1'} = (1 - \sigma) \cdot h_v^1 + \sigma \cdot h_{nn(v)}^1$$

where $\sigma \in [0, 1]$, and $Y_v^{1'} = Y_v$. 
Edge generator models the existence of edges among nodes and can predict the edges for the synthetic nodes.

- Trained on the real nodes and existing edges.
- Used to predict the neighbor information for the synthetic nodes.

\[ E_{v,u} = \text{softmax}(\sigma(h_v^1 \cdot S \cdot h_u^{1\top})) \]

where \( E_{v,u} \) predicts the relation between node \( u, v \) and \( S \) is the parameter matrix.

- Loss function:
  \[ L_{edge} = \|E - A\|_F^2 \]

- \( \tilde{A}[v', u] = \begin{cases} 
1, & \text{if } E_{v', u} > \eta \\
0, & \text{otherwise} 
\end{cases} \)
  or \( \tilde{A}[v', u] = E_{v', u} \).
GNN classifier

After adding the augmented nodes:

- Node representation: $H^1 \rightarrow \tilde{H}^1$
- Training set: $\mathcal{V}_L \rightarrow \tilde{\mathcal{V}}_L$
- Graph: $\tilde{\mathcal{G}} = \{\tilde{A}, \tilde{H}\}$

Introduce another block of GraphSAGE and a linear layer:

$$h^2_v = \sigma(W^2 \cdot \text{CONCAT}(h^1_v, \tilde{H}^1 \cdot \tilde{A}[:, v])),\]

$$P_v = \text{softmax}(\sigma(W^c \cdot \text{CONCAT}(h^2_v, H^2 \cdot \tilde{A}[:, v])),\]

where $H^2$ denotes the node representation from the second GraphSAGE block, $W^2, W^c$ refer to the weight parameters.

- Loss

$$L_{\text{node}} = - \sum_{u \in \tilde{\mathcal{V}}_L} \sum_c (1(Y_u == c) \cdot \log(P_v[c]))$$
Optimization Objective

Final objective function:

$$\min_{W^1, S, W^2, W^c} L_{node} + \lambda \cdot L_{edge}$$

Figure: Overview of the framework.
Experiments

- **Dataset:**
  1. **Cora:** citation network containing 2708 papers from 7 areas.
     - Majority classes: each class have a training set containing 20 nodes.
     - Minority class: randomly samples 3 classes and down-sample $20 \times \text{imbalance ratio}(0.5)$.
  2. **BlogCatalog:** 25%, 25%, 50% for training, validation and test set.
     - Majority:
     - Minority: 14 classes smaller than 100.
  3. **Twitter:** 25%, 25%, 50% for training, validation and test set.
     - Imbalanced ratio: 1:30.

- **Baselines:**
  - Over-sampling, Re-weight.
  - SMOTE, Embed-SMOTE.
  - $\text{GraphSMOTE}_T$, $\text{GraphSMOTE}_O$
  - $\text{GraphSMOTE}_{preT}$, $\text{GraphSMOTE}_{preO}$
## Experiments

### Figure: Comparison of different approaches for imbalanced node classification.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Cora</th>
<th></th>
<th>BlogCatalog</th>
<th></th>
<th>Twitter</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC</td>
<td>AUC-ROC</td>
<td>F Score</td>
<td>ACC</td>
<td>AUC-ROC</td>
<td>F Score</td>
</tr>
<tr>
<td>Origin</td>
<td>0.681±0.001</td>
<td>0.914±0.002</td>
<td>0.684±0.003</td>
<td>0.210±0.004</td>
<td>0.586±0.002</td>
<td>0.074±0.002</td>
</tr>
<tr>
<td>over-sampling</td>
<td>0.692±0.009</td>
<td>0.918±0.005</td>
<td>0.666±0.008</td>
<td>0.203±0.004</td>
<td>0.599±0.003</td>
<td>0.077±0.001</td>
</tr>
<tr>
<td>Re-weight</td>
<td>0.697±0.008</td>
<td>0.928±0.005</td>
<td>0.684±0.004</td>
<td>0.206±0.005</td>
<td>0.587±0.003</td>
<td>0.075±0.003</td>
</tr>
<tr>
<td>SMOTE</td>
<td>0.696±0.011</td>
<td>0.920±0.008</td>
<td>0.673±0.003</td>
<td>0.205±0.004</td>
<td>0.595±0.003</td>
<td>0.077±0.001</td>
</tr>
<tr>
<td>Embed-SMOTE</td>
<td>0.683±0.007</td>
<td>0.913±0.002</td>
<td>0.673±0.002</td>
<td>0.205±0.003</td>
<td>0.588±0.002</td>
<td>0.076±0.001</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;T&lt;/sub&gt;</td>
<td>0.713±0.008</td>
<td>0.929±0.006</td>
<td>0.720±0.002</td>
<td>0.206±0.005</td>
<td>0.602±0.004</td>
<td>0.083±0.003</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;O&lt;/sub&gt;</td>
<td>0.709±0.010</td>
<td>0.927±0.011</td>
<td>0.712±0.003</td>
<td>0.215±0.010</td>
<td>0.591±0.012</td>
<td>0.080±0.005</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;preT&lt;/sub&gt;</td>
<td>0.727±0.003</td>
<td>0.931±0.002</td>
<td>0.726±0.001</td>
<td>0.249±0.002</td>
<td>0.641±0.001</td>
<td>0.126±0.001</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;preO&lt;/sub&gt;</td>
<td>0.736±0.001</td>
<td>0.934±0.002</td>
<td>0.727±0.001</td>
<td>0.243±0.002</td>
<td>0.641±0.002</td>
<td>0.123±0.001</td>
</tr>
</tbody>
</table>
Experiments

Figure: Affects of over-sampling scale on Cora dataset.

Figure: Affects of hyper-parameter $\lambda$ on Cora dataset.
### Experiments

<table>
<thead>
<tr>
<th>Methods</th>
<th>Imbalance Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>Origin over-sampling</td>
<td>0.8681</td>
</tr>
<tr>
<td>Re-weight</td>
<td>0.8707</td>
</tr>
<tr>
<td>SMOTE</td>
<td>0.8791</td>
</tr>
<tr>
<td>Embed-SMOTE</td>
<td>0.8742</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;T&lt;/sub&gt;</td>
<td>0.8824</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;O&lt;/sub&gt;</td>
<td>0.8849</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;preT&lt;/sub&gt;</td>
<td>0.9167</td>
</tr>
<tr>
<td>GraphSMOTE&lt;sub&gt;preO&lt;/sub&gt;</td>
<td>0.9117</td>
</tr>
</tbody>
</table>

**Figure:** Node classification performance on Cora under various imbalance ratios.


Tianxiang Zhao, Xiang Zhang, and Suhang Wang. “GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks”. In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 833–841.